

106-2: EE4052

通識課程：

計算機程式設計

之旅

Computer Programming

Unit 07: 函數 - 計算與排序

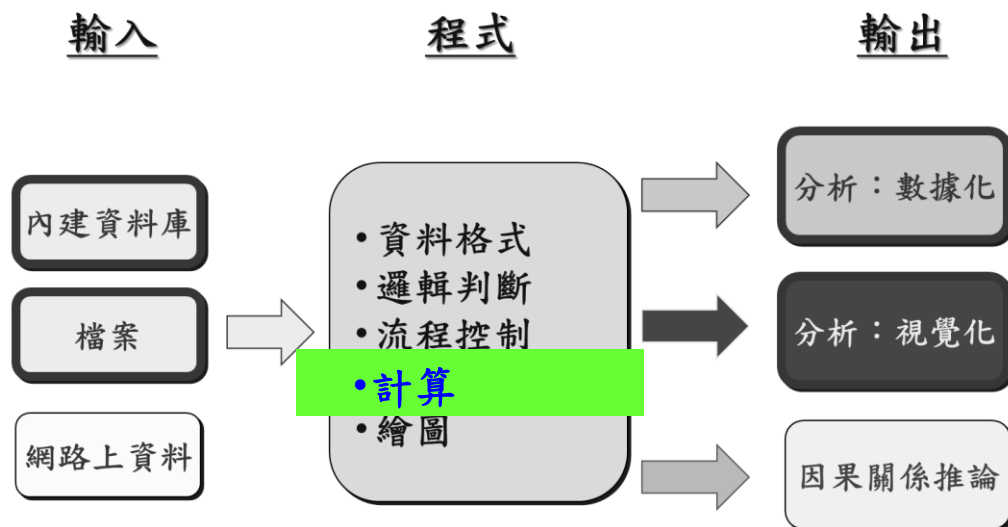
連 豐 力

臺大電機系

Feb 2018 - Jun 2018

課程主題進度

- **U01:** 課程介紹：討論主題，作業，報告，進行方式
- **U02:** 主題，案例，程式，演算法，資源
- **U03:** 設定軟體 R 與 Rstudio
- **U04:** 數據處理與繪圖指令功能
- **U05:** 資料類別與基本運算
- **U06:** 邏輯判斷與流程控制
- **U07:** 函數：計算與排序
- **U08:** 多維度資料格式
- **U09:** 檔案資料輸入與輸出
- **U10:** 繪圖功能與文字
- **U11:** 多重繪圖與顏色
- **U12:** 函數：動畫與動作
- **U13:** 探索性資料分析
- **U14:** 資料間的相關性
- **U15:** 資料連結分析



大綱

■ 常用函數

- 基本操作，基本統計，排序

■ 使用者自訂函數

- 符號函數：sign
- 計算總和，平均值，標準差
- 數據排序
- 找出：最大值，最小值
- 找出：中位數，四分位數

■ 自定數學函數

作業

HW05：函數 - 計算與排序

On 4/17, 2018

- 正規化 (Normalization) 的定義為：
 - The normalization of ratings means adjusting values measured on different scales to a notionally common scale, often prior to averaging.

- 正規化 (Normalization) 的公式為：

$$\frac{X - \bar{X}}{s}$$

\bar{X}

s

是平均值

是標準差

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i = \frac{1}{n} (x_1 + \cdots + x_n)$$

$$s = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2}$$

HW05：函數 - 計算與排序

On 4/17, 2018

- 請您寫一個函數 (function), `myNorm()`,
可以把一組數據進行正規化 (Normalization)
- 請同時用 `myNorm()` 與 `scale()` 兩個進行測試，看看結果是否一樣？
- 請測試比較下面幾個範例：
 - `myNorm(iris[, 1])`
 - `scale(iris[, 1])`
 - `myNorm(CO2[, 5])`
 - `scale(CO2[, 5])`
- 在 `.R` 的程式碼中，註解您所加註的程式碼的意義或想法。

HW05：函數 - 計算與排序

On 4/17, 2018

- 繳交下面檔案，檔案名稱：**HW05_學號_關鍵字.xxx**
 - 函數檔案：**HW05_B01921001_myNormFunc.R**
 - 測試程式檔案：**HW05_B01921001_myNormTest.R**
 - 報告檔案：**HW05_B01921001_myNormRpt.pdf**
- 繳交方式與期限：
 - E-mail 上面兩個檔案到：ntucp2018s@gmail.com
 - E-mail 主旨：**HW05 B01921001 myNorm**
(就是，作業編號 您的學號 關鍵字)
 - 繳交期限：**4/22 (Sun), 2018, 11pm 以前**
- 學習方式：請至下面網址輸入此次的學習方式所花的時間：
 - <https://goo.gl/k7tKLk>
 - https://docs.google.com/forms/d/e/1FAIpQLSdAZ_b-FUtvnNr_14rYQNYejMhDESy6jJ9ESh5XsjFI-DXMIw/viewform?c=0&w=1

常用函數

常用的函數 - 基本操作

- `x <- c(1, 8, 5, 2, 3, 1)`

- `length(x)`

- `diff(x)`

- `sum(x)`

- `prod(x)`

- `max(x)`

- `min(x)`

- `which.max(x)`

- `which.min(x)`

- `range(x)`

常用的函數 - 基本統計

- `x <- c(1, 8, 5, 2, 3, 1)`
- `round(x * pi, 2)`
- `cumsum(x)`
- `cumprod(x)`
- `unique(x)`
- `mean(x)`
- `median(x)`
- `var(x)`
- `sd(x)`
- `summary(x)`

常用的函數 - 排序

- `x <- c(1, -3, 5, -6, 0, 3)`

- `sort(x)`
- `rank(x)`
- `order(x)`

- `x[order(x)]`
- `order(x)[3]`

常用的函數 - 排序 - 逆向

- `x <- c(1, -3, 5, -6, 0, 3)`

- `which(rank(x) == 3)`
- `sort(x)[3]`
- `x[order(x)[3]]`

- `sort(x, decreasing = TRUE)`
- `rev(sort(x))`
- `rev(rank(x))`
- `order(x, decreasing = TRUE)`
- `rev(order(x))`

常用的函數 - 排序 - 平手

- `x <- c(2, 2, 2, 2, 1, 4, -2, 6, 6)`
- `sort(x)`
- `rank(x, ties.method = "average")`
- `rank(x, ties.method = "first")`
- `rank(x, ties.method = "random")`
- `rank(x, ties.method = "max")`
- `rank(x, ties.method = "min")`

- `rank(x)`

使用者自訂函數

定義一個函數

```

my_function_name <- function( arg_1, arg_2, ... ){
  statements
  return( object )
}
    
```

- `arg_1, arg_2, ..., :` argument, 引數, 輸入值
- `return:` 傳回 object 值 或傳回 object 資料

自定函數 - 符號函數

- $\text{sgn}(x) = -1, \quad x < 0$

- $0, \quad x = 0$

- $1, \quad x > 0$

- $\text{sgn}(2) \rightarrow 1$

- $\text{sgn}(-3) \rightarrow -1$

- $\text{sgn}(0) \rightarrow 0$

自定函數 - 符號函數

- $\text{sgn}(x) = \begin{matrix} -1, & x < 0 \\ 0, & x = 0 \\ 1, & x > 0 \end{matrix}$

- ```
mySign <- function(x) {
 if (x < 0) {
 value <- -1
 } else if (x == 0) {
 value <- 0
 } else {
 value <- 1
 }
 return(value)
}
```

# 自定函數 - 符號函數

- `mySign( 2 )`
- `mySign( -3 )`
- `mySign( 0 )`

# 自定函數 - 計算總和

- 給定一組數據，計算所有數據的總和

- 1, 2, 3, 4, 5

- $sum = 0$   $= (0)$

- $sum = (0) + 1$   $= (0 + 1)$

- $sum = (0 + 1) + 2$   $= (0 + 1 + 2)$

- $sum = (0 + 1 + 2) + 3$   $= (0 + 1 + 2 + 3)$

- $sum = (0 + 1 + 2 + 3) + 4$   $= (0 + 1 + 2 + 3 + 4)$

- $sum = (0 + 1 + 2 + 3 + 4) + 5$   $= (0 + 1 + 2 + 3 + 4 + 5)$

- $sum = (0 + 1 + 2 + 3 + 4 + 5)$

$$\sum_{i=1}^n x_i = (x_1 + \dots + x_n)$$

# 自定函數 - 計算總和

- 給定一組數據，計算所有數據的總和

```
mySum <- function(x) {
```

```
 Num <- length(x)
```

```
 temp_sum <- 0
```

```
 for(k in 1:Num) {
```

```
 temp_sum <- temp_sum + x[k]
```

```
 }
```

```
 return(temp_sum)
```

```
}
```

```
mySum(1:10)
```

```
sum(1:10)
```

$$\sum_{i=1}^n x_i = (x_1 + \dots + x_n)$$

# 自定函數 - 計算平均值

- 給定一組數據，計算所有數據的**平均值**

```

myMean <- function(x) {
 Num <- length(x)
 temp_sum <- 0
 for(k in 1:Num) {
 temp_sum <- temp_sum + x[k]
 }
 return(temp_sum / Num)
}

```

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i = \frac{1}{n} (x_1 + \dots + x_n)$$

- myMean ( 1:10 )
- mean( 1:10 )

# 自定函數 - 計算平均值

- 給定一組數據，計算所有數據的**平均值**

```
■ myMean <- function(x) {
 temp_Num <- length(x)
 temp_Sum <- mySum(x)
 return(temp_Sum / temp_Num)
}
```

- myMean ( 1:10 )
- mean( 1:10 )

# 自定函數 - 計算標準差

- 給定一組數據，計算所有數據的標準差
- 母體的標準差

$$\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^N (X_i - \mu)^2}$$

- 樣本的標準差

$$s = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2}$$

- Data: `x[1], x[2], x[3], ...`
- mean: `x_mean <- myMean( x )`
- diff:
  - `x_diff_1[ i ] <- x[ i ] - x_mean`
  - `x_diff_2[ i ] <- ( x_diff_1[ i ] )^2`
  - `x_diff_3[ i ] <- ( x[ i ] - x_mean )^2`
- sum: `x_sum <- mySum( x_diff_2 )`  
`x_sum <- mySum( x_diff_3 )`
- sqrt: `sqrt ( x_sum / ( N - 1 ) )`

# 自定函數 - 計算標準差

- 給定一組數據，計算所有數據的標準差

- `mySD <- function( x ) {`

- `Num <- length( x )`

- `temp_sum <- 0`

- `temp_mean <- 0`

- `temp_diff <- rep(0, length.out = Num)`

- `for( k in 1:Num ) {`

- `temp_sum <- temp_sum + x[ k ]`

- `}`

- `temp_mean <- temp_sum/Num`

- `for( k in 1:Num ) {`

- `temp_diff[ k ] <- ( x[ k ] - temp_mean )^2`

- `}`

- `temp_sum_diff <- mySum( temp_diff )`

- `temp_SD <- sqrt( temp_sum_diff / ( Num - 1 ) )`

- `return( temp_SD )`

- `}`

- `mySD( 1:10 )`

- `sd( 1:10 )`



# 自定函數 - 計算標準差

```
■ mySD <- function(x) {
 Num <- length(x)
 temp_sum <- 0
 temp_mean <- 0
 temp_diff <- rep(0, length.out = Num)
 for(k in 1:Num) {
 temp_sum <- temp_sum + x[k]
 }
 temp_mean <- temp_sum/Num
 print(temp_mean)
 for(k in 1:Num) {
 temp_diff[k] <- (x[k] - temp_mean)^2
 }
 print(temp_diff)
 temp_sum_diff <- mySum(temp_diff)
 print(temp_sum_diff)
 temp_SD <- sqrt(temp_sum_diff / (Num - 1))
 return(temp_SD)
}
```

```
■ mySD(1:10)
■ sd(1:10)
```

# 自定函數 - 數據排序

- 給定一組數據，按照大小順序排序

```
> mySort(c(3, 5, 2, 1))
```

```
[1] 1 2 3 5
```

```
> sort(c(3, 5, 2, 1))
```

```
[1] 1 2 3 5
```

```
> mySort(c(-3, 5, 2, 1, -2, 4))
```

```
[1] -3 -2 1 2 4 5
```

```
> sort(c(-3, 5, 2, 1, -2, 4))
```

```
[1] -3 -2 1 2 4 5
```

# 自定函數 - 數據排序

- 0 9 8 7 6 5 4 3 2 1
- 1 9 8 7 6 5 4 3 2 1
- 1 9 8 7 6 5 4 3 2 1
- 1 8 9 7 6 5 4 3 2 1
- 2 8 9 7 6 5 4 3 2 1
- 2 8 9 7 6 5 4 3 2 1
- 2 8 7 9 6 5 4 3 2 1
- 3 8 7 9 6 5 4 3 2 1
- 3 8 7 9 6 5 4 3 2 1
- 3 8 7 6 9 5 4 3 2 1
- 4 8 7 6 9 5 4 3 2 1
- 4 8 7 6 9 5 4 3 2 1
- 4 8 7 6 5 9 4 3 2 1
- 5 8 7 6 5 9 4 3 2 1
- 5 8 7 6 5 9 4 3 2 1
- 5 8 7 6 5 4 9 3 2 1
- 6 8 7 6 5 4 9 3 2 1
- 6 8 7 6 5 4 9 3 2 1
- 6 8 7 6 5 4 3 9 2 1
- 7 8 7 6 5 4 3 9 2 1
- 7 8 7 6 5 4 3 9 2 1
- 7 8 7 6 5 4 3 2 9 1
- 8 8 7 6 5 4 3 2 9 1
- 8 8 7 6 5 4 3 2 9 1
- 8 8 7 6 5 4 3 2 1 9
- 9 8 7 6 5 4 3 2 1 9

# 自定函數 - 數據排序

■ 0 8 7 6 5 4 3 2 1 9

■ 1 8 7 6 5 4 3 2 1 9

■ 1 8 7 6 5 4 3 2 1 9

■ 1 7 8 6 5 4 3 2 1 9

■ 2 7 8 6 5 4 3 2 1 9

■ 2 7 8 6 5 4 3 2 1 9

■ 2 7 6 8 5 4 3 2 1 9

■ 3 7 6 8 5 4 3 2 1 9

■ 3 7 6 8 5 4 3 2 1 9

■ 3 7 6 5 8 4 3 2 1 9

■ 4 7 6 5 8 4 3 2 1 9

■ 4 7 6 5 8 4 3 2 1 9

■ 4 7 6 5 4 8 3 2 1 9

■ 5 7 6 5 4 8 3 2 1 9

■ 5 7 6 5 4 8 3 2 1 9

■ 5 7 6 5 4 3 8 2 1 9

■ 6 7 6 5 4 3 8 2 1 9

■ 6 7 6 5 4 3 8 2 1 9

■ 6 7 6 5 4 3 2 8 1 9

■ 7 7 6 5 4 3 2 8 1 9

■ 7 7 6 5 4 3 2 8 1 9

■ 7 7 6 5 4 3 2 1 8 9

■ 8 7 6 5 4 3 2 1 8 9

# 自定函數 - 數據排序

■ 0 8 7 6 5 4 3 2 1 9

■ ...

■ 8 7 6 5 4 3 2 1 8 9

■ 0 7 6 5 4 3 2 1 8 9

■ ...

■ 7 6 5 4 3 2 1 7 8 9

■ 0 6 5 4 3 2 1 7 8 9

■ ...

■ 6 5 4 3 2 1 6 7 8 9

■ 0 5 4 3 2 1 6 7 8 9

■ ...

■ 5 4 3 2 1 5 6 7 8 9

■ 0 4 3 2 1 5 6 7 8 9

■ ...

■ 4 3 2 1 4 5 6 7 8 9

■ 0 3 2 1 4 5 6 7 8 9

■ ...

■ 3 2 1 3 4 5 6 7 8 9

■ 0 2 1 3 4 5 6 7 8 9

■ ...

■ 2 1 2 3 4 5 6 7 8 9

# 自定函數 - 數據排序

■ 0 3 5 1 2 4

■ 2 3 5 1 2 4

■ 2 3 1 5 2 4

■ 3 3 1 5 2 4

■ 3 3 1 2 5 4

■ 4 3 1 2 5 4

■ 4 3 1 2 4 5

■ 0 3 1 2 4 5

■ 1 3 1 2 4 5

■ 1 1 3 2 4 5

■ 2 1 3 2 4 5

■ 2 1 2 3 4 5

■ 0 3 2 5 4 1

■ 1 3 2 5 4 1

■ 1 2 3 5 4 1

■ 3 2 3 5 4 1

■ 3 2 3 4 5 1

■ 4 2 3 4 5 1

■ 4 2 3 4 1 5

■ 3 2 3 4 1 5

■ 3 2 3 1 4 5

■ 2 2 3 1 4 5

■ 2 2 1 3 4 5

■ 1 2 1 3 4 5

■ 1 1 2 3 4 5

# 自定函數 - 數據排序

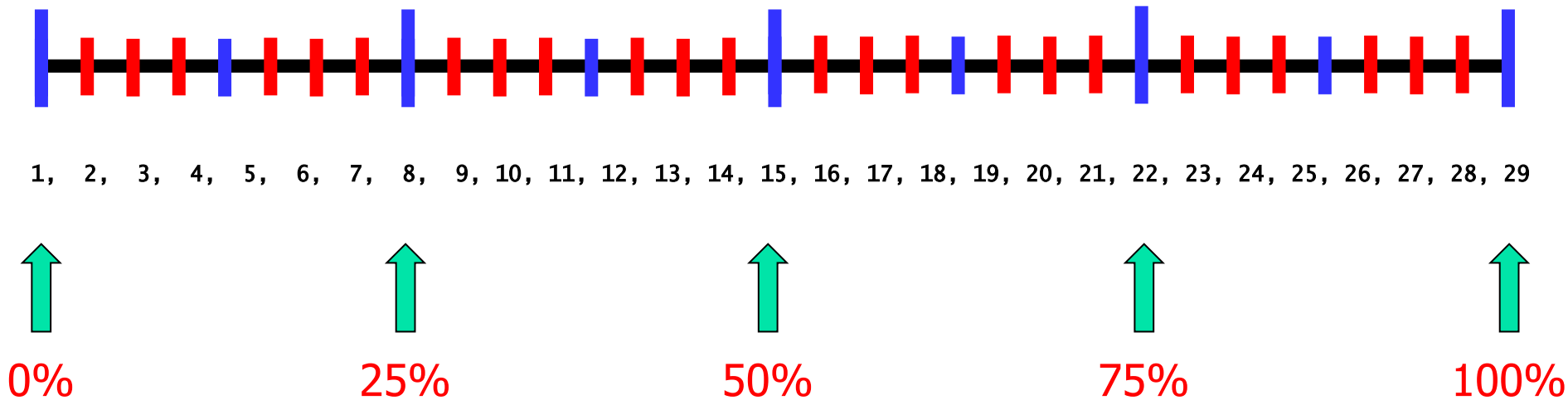
```
■ mySort <- function(x) {
 itemCount <- length(x)
 repeat {
 hasChanged <- FALSE
 itemCount <- itemCount - 1
 if (itemCount >= 1){
 for(k in 1 : itemCount) {
 if (x[k] > x[k+1]) {
 t <- x[k]
 x[k] <- x[k+1]
 x[k+1] <- t
 hasChanged <- TRUE
 }
 }
 }
 if (!hasChanged) break;
 }
 return(x)
}
```

# 自定函數 - 最大值，最小值

- `myMax` <- function( x ) {  
    Num <- length( x )  
    temp <- mySort( x )  
    return( temp[ Num ] )  
}
- `myMin` <- function( x ) {  
    Num <- length( x )  
    temp <- mySort( x )  
    return( temp[ 1 ] )  
}
- `myRange` <- function( x ) {  
    temp\_max <- myMax( x )  
    temp\_min <- myMin( x )  
    temp <- c( temp\_min, temp\_max )  
    return( temp )  
}



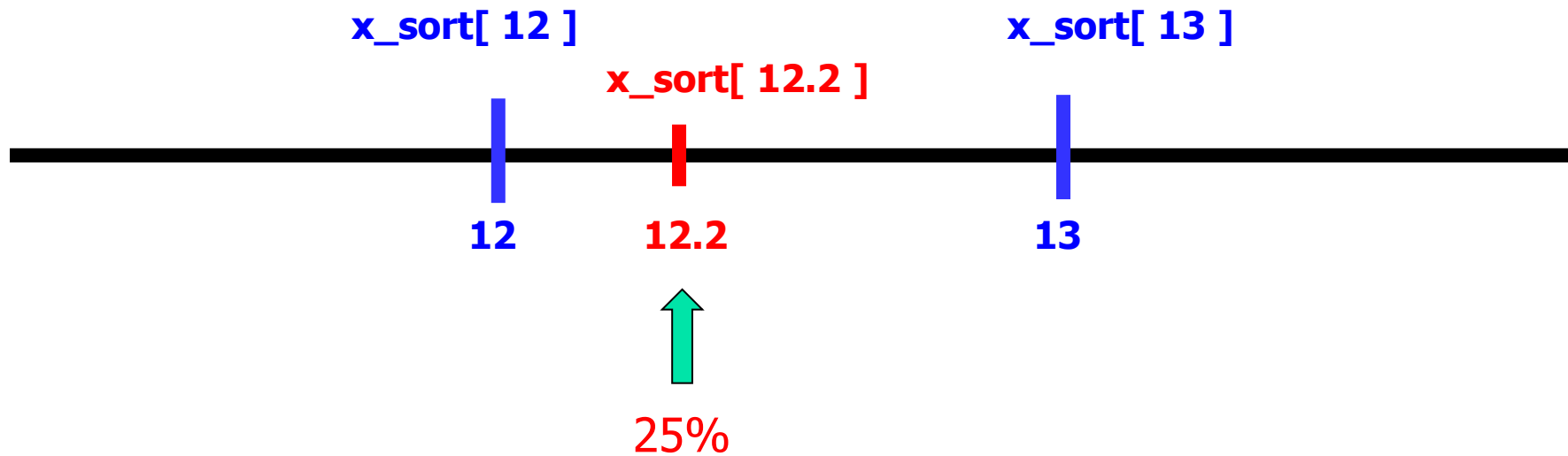
# 自定函數 - 中位數，四位數



- 如果有101個數據，也就是：100個間隔
- 50% 即是第51個數據，25% 即是第26個數據，75% 即是第76個數據
- 如果不是剛好的個數，
  - 例如：25% 是第 12.2 個數據
  - 則這個 25% 的數據會是：第 12 個數據與第 13 個數據的比例分配計算，
  - 也就是：

# 自定函數 - 中位數，四位數

- 也就是：



$x\_sort[25\%] =$

$$x\_sort[12.2] = x\_sort[12] + \frac{(x\_sort[13] - x\_sort[12])}{13 - 12} * (12.2 - 12)$$

# 自定函數 - 中位數，median

```
■ myMedian <- function(x) {
 Num <- length(x)
 x_sort <- mySort(x)
 if((Num-1)/2 == floor((Num-1)/2)){
 temp_median <- x_sort[(Num-1)/2+1]
 }
 else {
 temp <- floor(Num/2)
 temp_median <- (x_sort[temp] + x_sort[temp+1])/2
 }
 return(temp_median)
}
```

# 自定函數 - 四位數 (25%, 75%)

```
■ my25p <- function(x) {
 tp_num <- length(x)
 x_sort <- mySort(x)
 tp_max <- myMax(x)
 tp_min <- myMin(x)
 tp1 <- floor ((tp_num-1) * 0.25) +1
 tp2 <- ((tp_num-1) * 0.25) +1
 tp3 <- ceiling ((tp_num-1) * 0.25) +1

 if(tp2 == tp1){
 tp_25p <- x_sort[tp1]
 }
 else {

tp_25p <- x_sort[tp1] + ((tp2-tp1)/(tp3-tp1))*(x_sort[tp3]-x_sort[tp1])

 }
 return(tp_25p)
}
```

# 自定函數 - 四位數 (25%, 75%)

```
■ my75p <- function(x) {
 tp_num <- length(x)
 x_sort <- mySort(x)
 tp_max <- myMax(x)
 tp_min <- myMin(x)
 tp1 <- floor ((tp_num-1) * 0.75) +1
 tp2 <- ((tp_num-1) * 0.75) +1
 tp3 <- ceiling ((tp_num-1) * 0.75) +1

 if(tp2 == tp1){
 tp_75p <- x_sort[tp1]
 }
 else {

tp_75p <- x_sort[tp1] + ((tp2-tp1)/(tp3-tp1))*(x_sort[tp3]-x_sort[tp1])

 }
 return(tp_75p)
}
```

# 自定函數 – mySummary

```
■ mySummary <- function(x) {
```

```
 tp_summary <- c(myMin(x), my25p(x), myMedian(x), myMean(x),
 my75p(x), myMax(x))
```

```
 return(tp_summary)
```

```
}
```

# 自訂數學函數

# 自定數學函數 - $f(x)$

- $f(x) = x^3 * \cos(x) - 2 * x^2 * \sin(x) + 5 * x - 1$

- `fx <- function(x) {`

`x^3 * cos(x) - 2 * x^2 * sin(x) + 5 * x - 1`

`}`

- `fx( -5 )`

- `fx( 0 )`

- `fx( 5 )`

- `fx( c( -5, 0, 5 ) )`



# 自定數學函數 - $f(x, y)$

- $f(x, y) = x^3 * \cos(y) - 2 * x^2 * \sin(y) + 5 * x - 1$

- `fxxy <- function( x, y ) {`

  - `$x^3 * \cos(y) - 2 * x^2 * \sin(y) + 5 * x - 1$`

- `}`

- `fxxy( -5, pi )`

- `fxxy( 1, pi )`

- `fxxy( 1, pi/2 )`

- `fxxy( 0, pi/6 )`

# 自定數學函數 - $f(x, y, z)$

- $f(x, y, z) = x^3 * \cos(y) - 2 * x^2 * \sin(z) + 5 * x - 1$

- `fxyz <- function( x, y, z ){`

  - `$x^3 * \cos(y) - 2 * x^2 * \sin(z) + 5 * x - 1$`

  - `}`

- `fxyz( -5, pi, pi )`

- `fxyz( 1, pi, pi/2 )`

- `fxyz( 1, pi/2, pi/2 )`

- `fxyz( 0, pi/6, 3*pi )`

下課了